Create an image for the latest backup

One key use case for SQL Clone is to have the latest possible copy of a production database available for quick mounting in another environment.

For a backup file matching a database name, this script obtains the latest then uses it to create a SQL Clone image.

> You need to provide a SQL Clone Agent to perform the operation, but you only need the disk space for the database to be available in the *final* destination (the operation uses a virtual mount point into that location).

```
# Script to create a new SQL Clone data image from a backup file

$SQLCloneServer= 'http://sql-clone.example.com:14145'
$SQLCloneAgent = 'wks-dev1'

Connect-SqlClone -ServerUrl $SQLCloneServer

$SourceDatabase = 'Forex'
$BackupFolder = '\\File1.example.com\Backups\SQL\MSSQL\Backup' # a folder which has been previously used in SQL
Clone (thereby registered)

if (!(Test-Path ($BackupFolder)))
  {
    write-host 'Backup folder not found. Exiting.'
    break
  }

# Get the latest backup file for our database (striped backups would be more complex)
$BackupFiles = Get-ChildItem -Path $BackupFolder  |
    Where-Object -FilterScript { $_.Name.Substring(0,$SourceDatabase.Length) -eq $SourceDatabase}   # My backup
files always start with the database name

# Now we have a filtered list, sort to get latest
$BackupFile = $BackupFiles |
    Sort-Object -Property LastWriteTime  |
    Select-Object -Last 1 # I only want the most recent file for this database to be used

$BackupFileName = $BackupFile.Name

#Start a timer
$elapsed = [System.Diagnostics.Stopwatch]::StartNew()

"Started at {0}, creating data image for database ""{1}"" from backup file ""{2}""" -f $(get-date) ,
$SourceDatabase , $BackupFileName

$DataImageName = $SourceDatabase + "_" + (Get-Date -Format "yyyyMMdd") # Prepare a name for the data image,
with a timestamp
$ImageDestination = Get-SqlCloneImageLocation -Path '\\filestore.example.com\SQLClone\SQL Clone Images' # Point
to the file share we want to use to store the image

$CloneBackupLocation = Get-SqlCloneBackupLocation -Path $BackupFolder # Point to the backup folder we want to
work with (this was 'registered' with SQL Clone when I used the UI above)

$NewImage = New-SqlCloneImage -Name $DataImageName -BackupLocation $CloneBackupLocation -BackupFileName
$BackupFileName -Destination $ImageDestination | Wait-SqlCloneOperation # Create the data image and wait for
completion

"Total Elapsed Time: {0}" -f $($elapsed.Elapsed.ToString())
```